SCR.1.2 TD 07 ⊥ :

Les expressions régulières

I. Introduction.

Une *expression régulière exp* est un motif utilisant éventuellement des caractères spéciaux et qui définit un ensemble E_{exp} de chaînes de caractères. On dira pour une chaîne de caractères appartenant à E_{exp} qu'elle est en *correspondance* avec l'expression régulière *exp* (*matched* en anglais). Une expression régulière sert à caractériser par une seule description un ensemble de chaînes de caractères qui, sans cela, serait impossible de lister exhaustivement. Comment exprimer, par exemple, l'ensemble de toutes les chaînes de caractères contenant une suite d'au moins un `'b'` ? L'expression régulière `b+` fait un match avec n'importe quel élément de la forme `<a_non_null_sequence_of_b's>`. Autre exemple : l'ensemble de toutes les chaînes de caractères contenant une suite d'au moins un `'b'` précédée de 0 ou 1 fois `'a'` est décrit par l'expression régulière `a?b+`. Ainsi, dans une expression régulière, le caractère `?` indique la répétition 0 ou 1 fois (présence facultative) de ce qui précède le symbole `'?'`. Ici, `?` n'exprime pas "n'importe quel caractère" comme dans le mécanisme du globbing étudié il y a quelques TD/TP de cela. Ainsi, selon le contexte, les caractères spéciaux s'interprètent différemment.

Certaines commandes comprennent les expressions régulières parmi elles : `awk`, `grep`, `sed`. On connaît déjà les commandes `awk` et `sed`. La commande `grep` recherche les lignes du fichier en entrée qui contiennent une correspondance avec un motif donné sous la forme d'une expression régulière. La commande `grep` travaille sur l'entrée standard si aucun nom de fichier n'est spécifié. Par exemple, supposons qu'on veuille rechercher dans un fichier `alert_records`, s'il y a des lignes contenant plus de deux "ALERT". On pourra alors utiliser une expression régulière pour décrire la condition sur les chaînes "cibles", et passer cette expression régulière à la commande `grep` :

```
grep -E "(ALERT.*){3,}" alert_records
```

II. Une première liste d'expressions régulières.

- Le symbole `.` (period) coïncide avec n'importe quel caractère, exactement un caractère.
- Une liste de caractères entre `[ ]` est une expression régulière qui coïncide avec n'importe quel caractère -exactement un caractère- inclus dans les `[ ]`. Si le premier caractère de cette liste est `^` alors l'expression régulière coïncide avec n'importe quel caractère -exactement un caractère- qui ne soit pas inclus dans la liste. Pour définir un intervalle, on utilise l'expression régulière `[...-...]`
- Les symboles `^` et `$` coïncident respectivement avec le début et la fin de ligne.

Exercice 1. Quelles chaînes de caractères cherche-t-on à "attraper" avec les expressions régulières suivantes ? En donner des exemples.

```
(1) '^$'
(2) '^...$'
(3) '.c$'
(4) '.[config]$'
(5) '^..[[:digit:]]'
(6) '^.\.[[:digit:]]'
(7) '[^[:upper:]].$'
```

III. La répétition dans les expressions régulières.

```
A regular expression may be followed by one of several repetition operators:
  ?     The preceding item is optional and matched at most once.
  *     The preceding item will be matched zero or more times.
  +     The preceding item will be matched one or more times.
  {n}   The preceding item is matched exactly n times.
  {n,}  The preceding item is matched n or more times.
  {,m}  The preceding item is matched at most m times. This is a GNU extension.
  {n,m} The preceding item is matched at least n times, but not more than m times.
```

Exercice 2. Quelles chaînes de caractères cherche-t-on à "attraper" avec les expressions régulières suivantes ?

```
(1) 'T?ST?AR'
(2) '(back)?(slash)?'
(3) '^[[:space:]]*$'
(4) '^.?..$' , '^..?.$' , '^...?$' , '^.{2,3}$
(5) '(ding-deng-dong)+'
(6) '^[[:space:]]+$'
(7) '[$][[:digit:]]?$'
```

IV. L'alternance.

```
Alternation
  Two regular expressions may be joined by the infix  operator |;  the resulting
  regular expression matches any string matching either alternate expression.
```

Exercice 3. Même question qu'à l'exercice précédent.

```
(1) '[.](config|cfg)'
(2) '[$]([[:digit:]]|$)'
```

V. Référence arrière.

```
Back References and Subexpressions
 The back-reference \n, where n is a single digit, matches the substring previously
 matched by the nth parenthesized subexpression of the regular expression.
```

Exercice 4. Donner des exemples de chaînes de caractères qui sont en correspondance avec :

```
(1) '(.{3})(.{5})\2\1'
(2) '(.{3}(.{5}))\2\1'
```